Using Modeling and Simulation to Predict and Evaluate System Performance Throughout the Lifecycle

Robert A. Basta
Senior Systems Engineer
Harris Corporation
Government Communications Systems Division
Melbourne, Florida

Copyright, 2003 Harris Corporation, All Rights Reserved

6th Annual Systems Engineering Conference
Oct 20-23, 2003

Session: Modeling and Simulation
in Systems Engineering

rbasta@harris.com

Abstract

Successful completion of many DoD and other government systems is highly dependent on performance. A process to predict system timing and sizing performance during the early development phases, and to later measure how well components perform as they are developed, as well as later during system operation, is essential for success. This presentation will describe how Harris has successfully utilized this process on program efforts by covering:

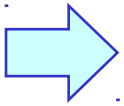
Modeling and Simulation: System modeling and simulation is a risk reduction activity to demonstrate the ability to meet or exceed contractual system level performance requirements. Since low-level implementation changes rarely correct performance problems, the need exists to evaluate and make architectural decisions and tests at the system level. Building the system first and then trying to improve performance can be difficult, risky, and costly.

System Trades: Modeling and simulation provides a means to make system decisions. Scenarios can be tested and tradeoffs evaluated for choosing an optimal solution. Trade efforts need to include all system segments since capacity, performance behavior, and cost are influenced by the interaction of the end-to-end system components. As requirements or designs change, simulation allows the trades to be quickly updated for continually providing an optimal system that meets requirements at an effective cost.

Accurate Predictions: The system modeling and simulation effort provides high confidence capacity sizing when backed up with subset functional benchmark testing. This allows verification of critical performance parameters within full-scale simulation models. In addition, it may be necessary to develop prototype code that emulates resource consumption of the full-up system to uncover unforeseen interactions that are not obtainable from functional testing.

Life Cycle Management/Control: Since systems are usually baselined at a fixed resource capacity that optimizes meeting performance requirements and overall system cost, resource consumption is budgeted and managed throughout the program life. Deltas are analyzed using the simulation model to determine impact. This process not only forces early issue resolution, but also the SW organization to be cognizant of performance and budget allocations.

Agenda

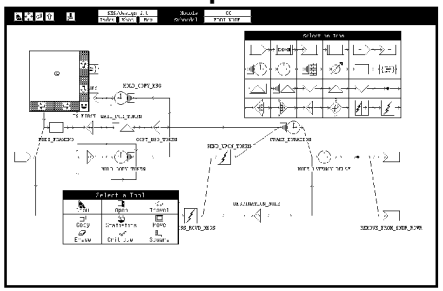


- **Modeling and Simulation**
- **System Trades**
- **Accurate Predictions**
- **Lifecycle Management / Control**
- **Lessons Learned**

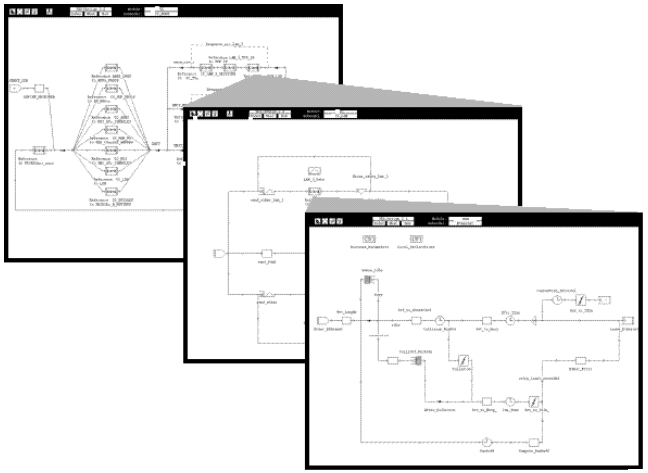
Features of an Effective Modeling and Simulation Tool

HARRIS

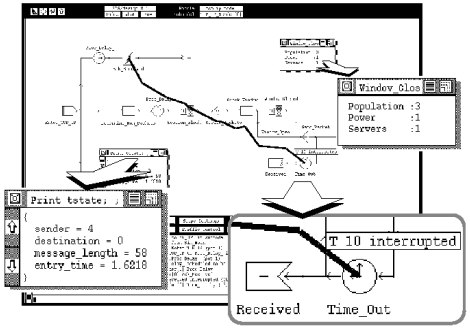
Graphical



Hierarchical



Animation & Run-Time I/F

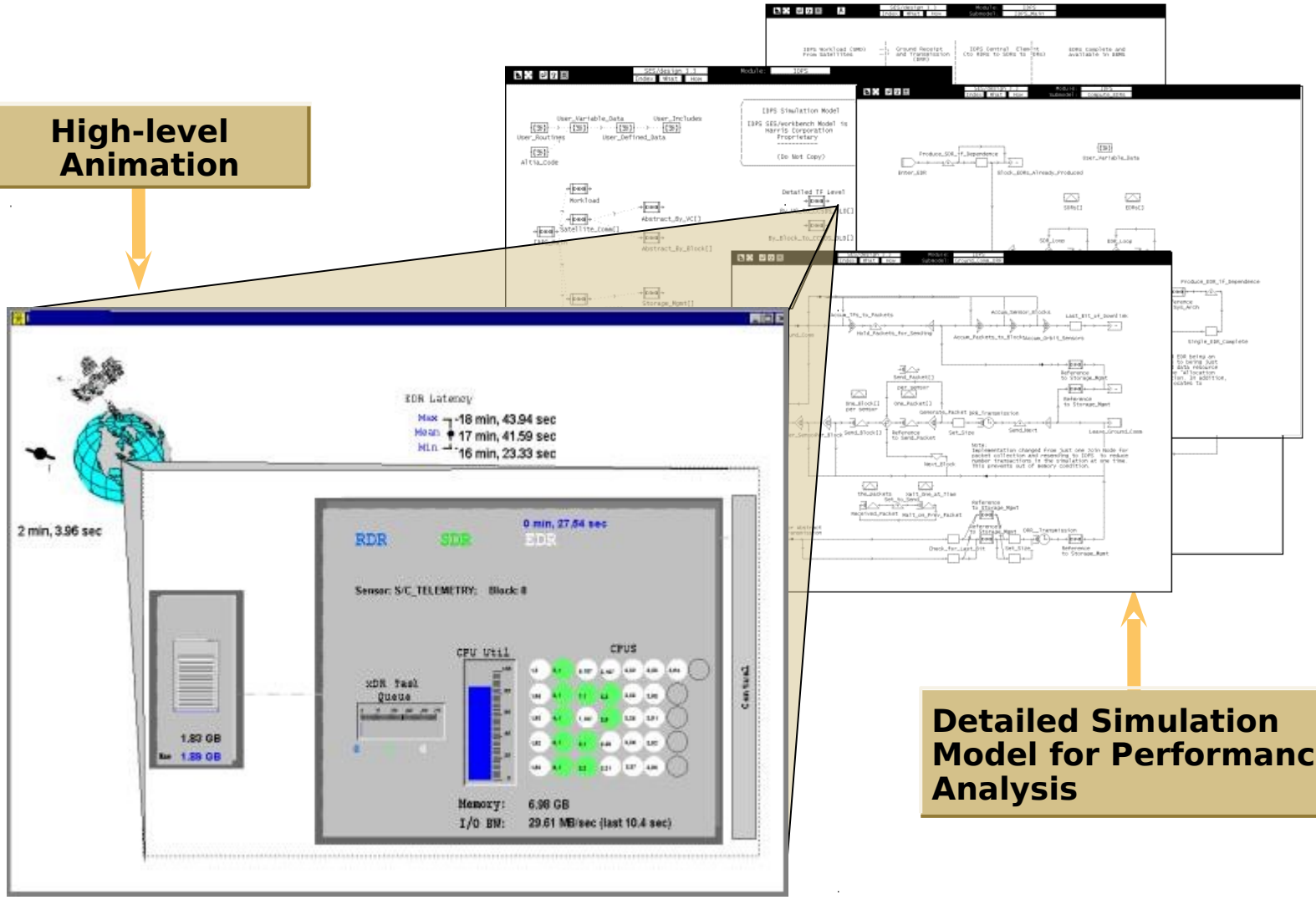


Procedural Char. (Code)

- **Graphical** environment provides effective means to build and communicate models
- **Hierarchical** Modeling allows for design refinement as details become known
- **Procedural characterization** provides accurate modeling of specific system behavior
- **Animation** is an integral part in verifying, debugging, and communicating models
- **Run-Time Interface**

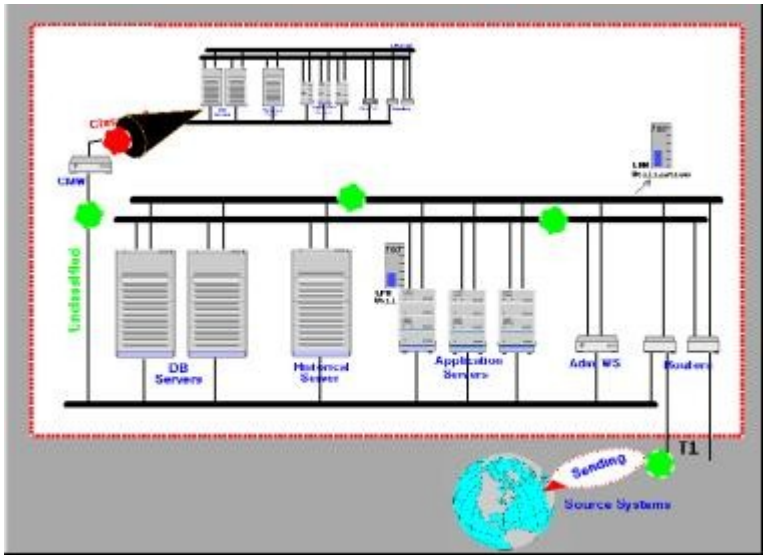
Toolset should Communicate System Behavior at a Higher Level of Abstraction

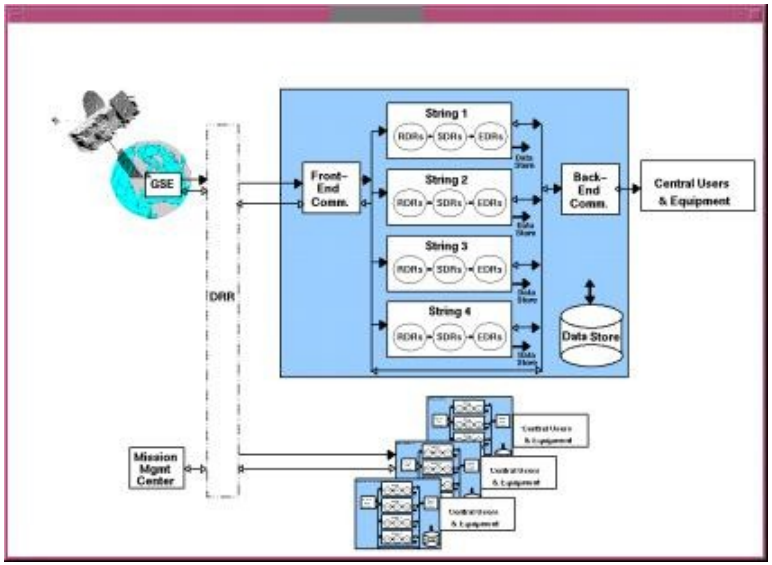
High-level Animation

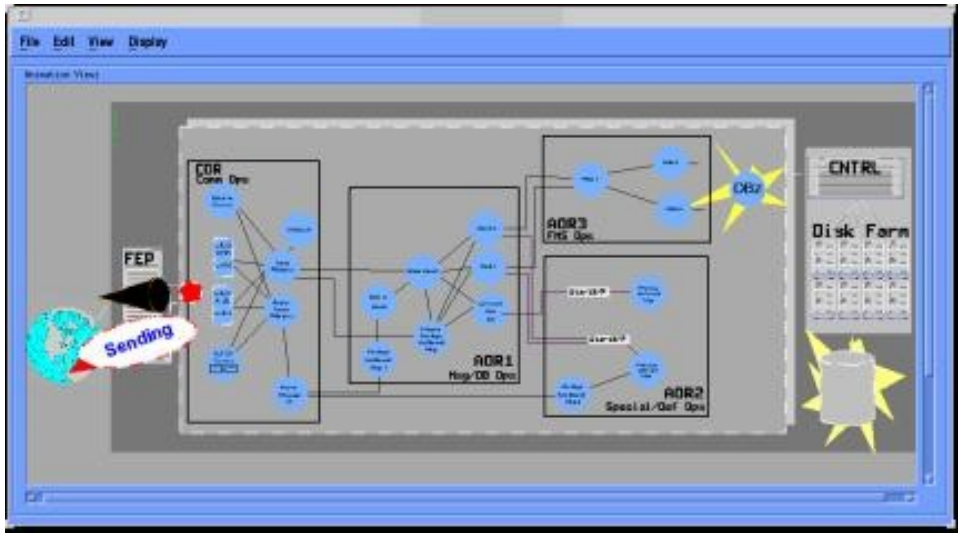


Detailed Simulation Model for Performance Analysis

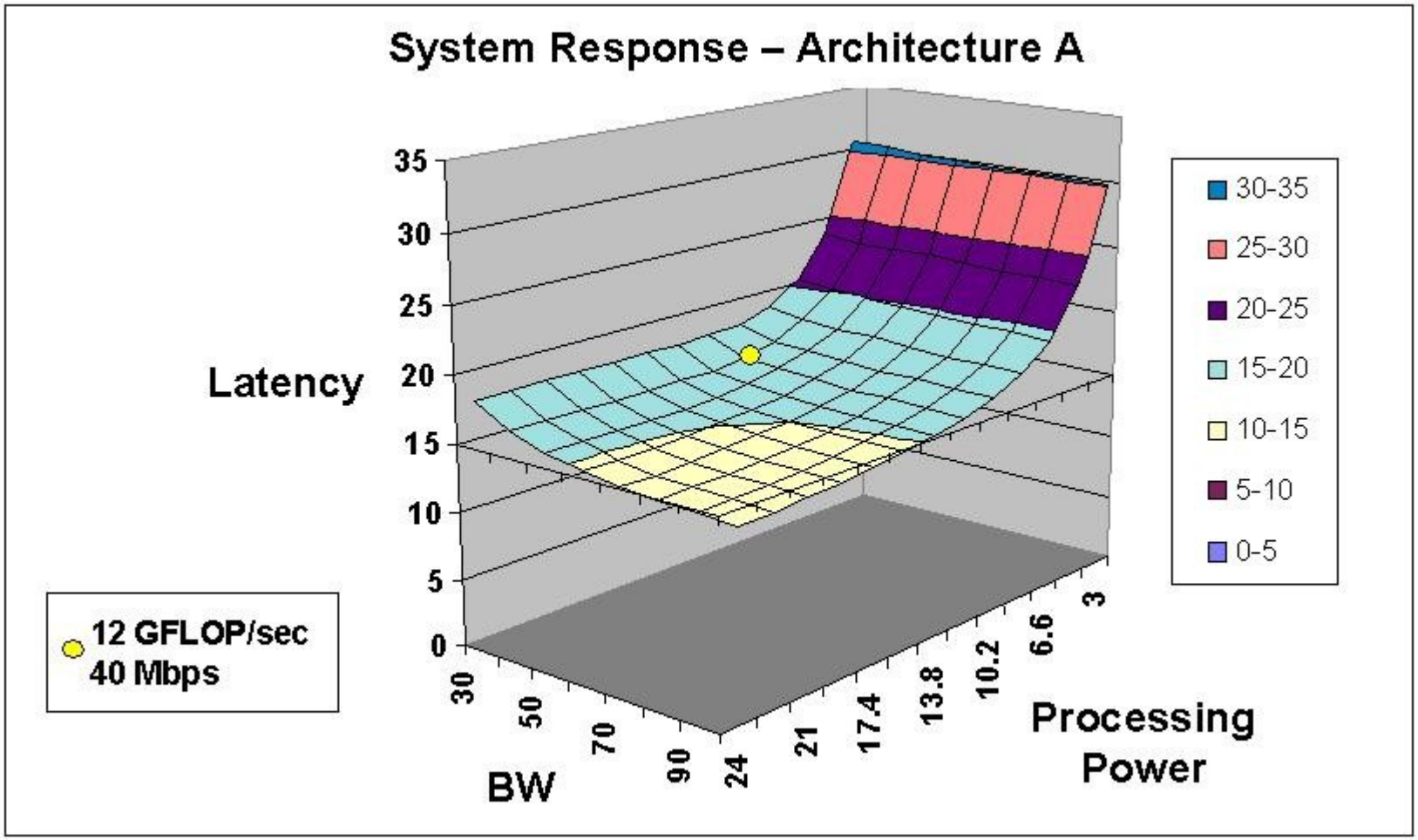
Provide Customer a Representative and Dynamic View of Their End-to-End System



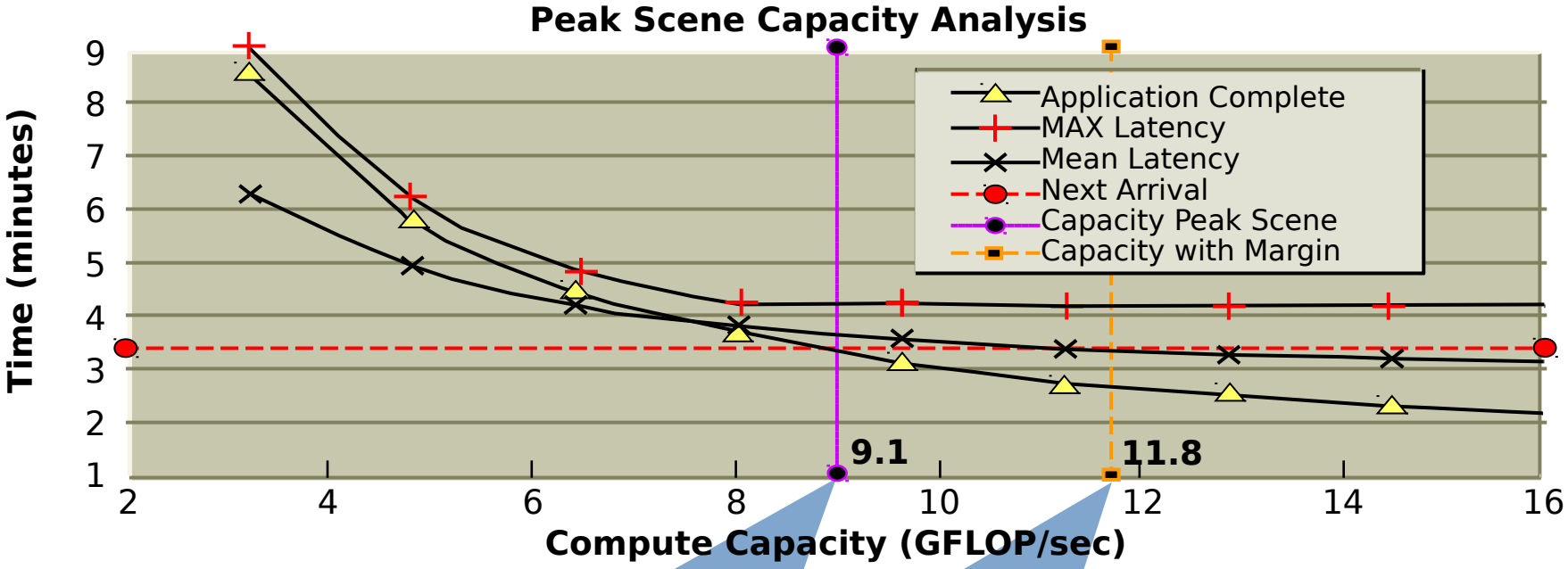




Understanding Performance Behavior is Critical when Designing Complex Systems

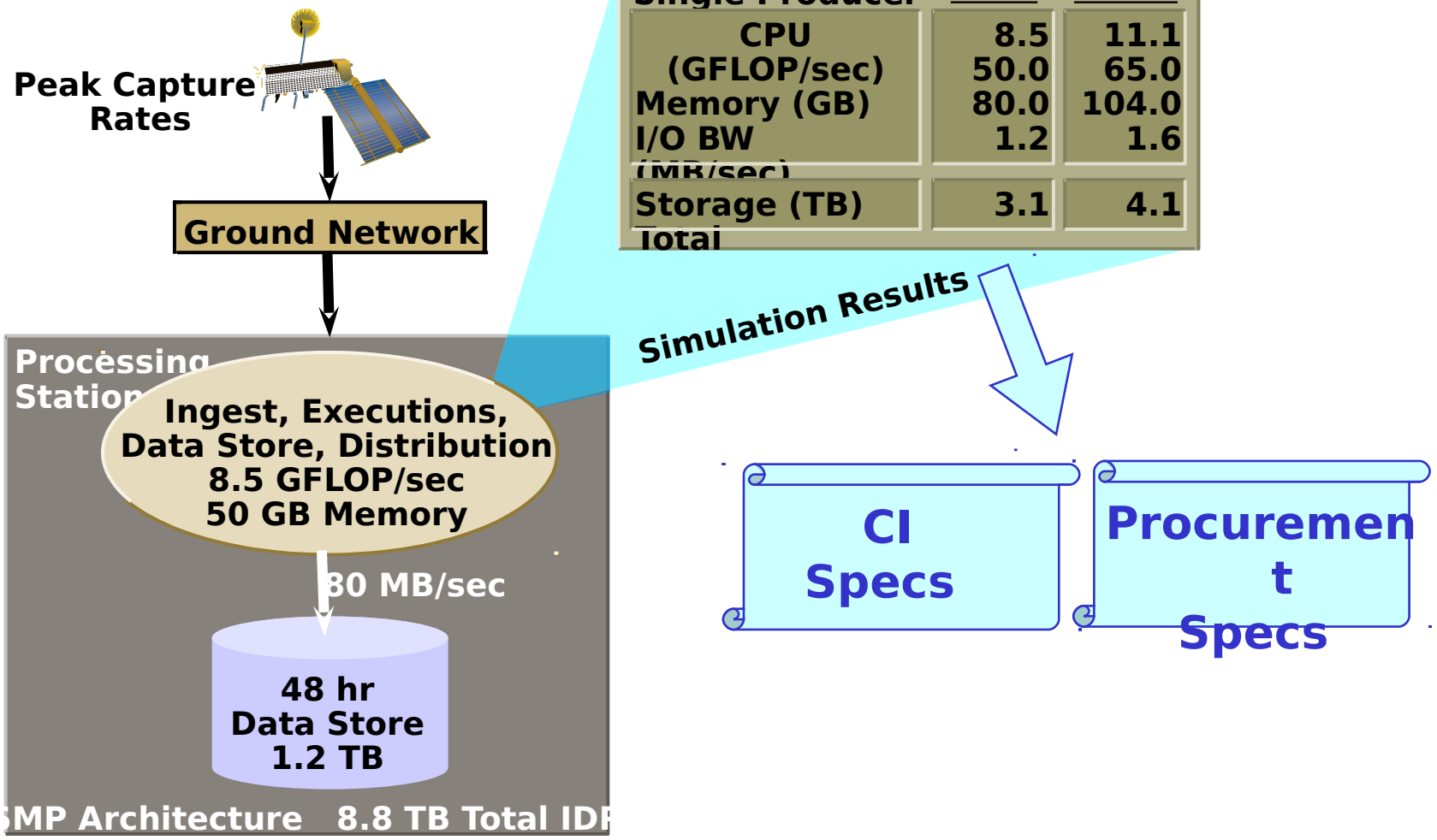


Demonstrate Ability to Meet or Exceed Contractual Performance Requirements



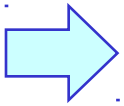
Results	Peak Scene Analysis		Average Scene Analysis	
	At 9.1 GFLOP/s	At 11.8 GFLOP/s	At 9.1 GFLOP/s	At 11.8 GFLOP/s
Application	3.5	2.8	2.9	2.4
Complete	1.5	1.5	1.2	1.2
Secondary	4.1	4.1	3.9	3.9
Complete	70%	66%	65%	60%
Max Latency				
CPU Utilization				

Establish the Requirements to Size the System



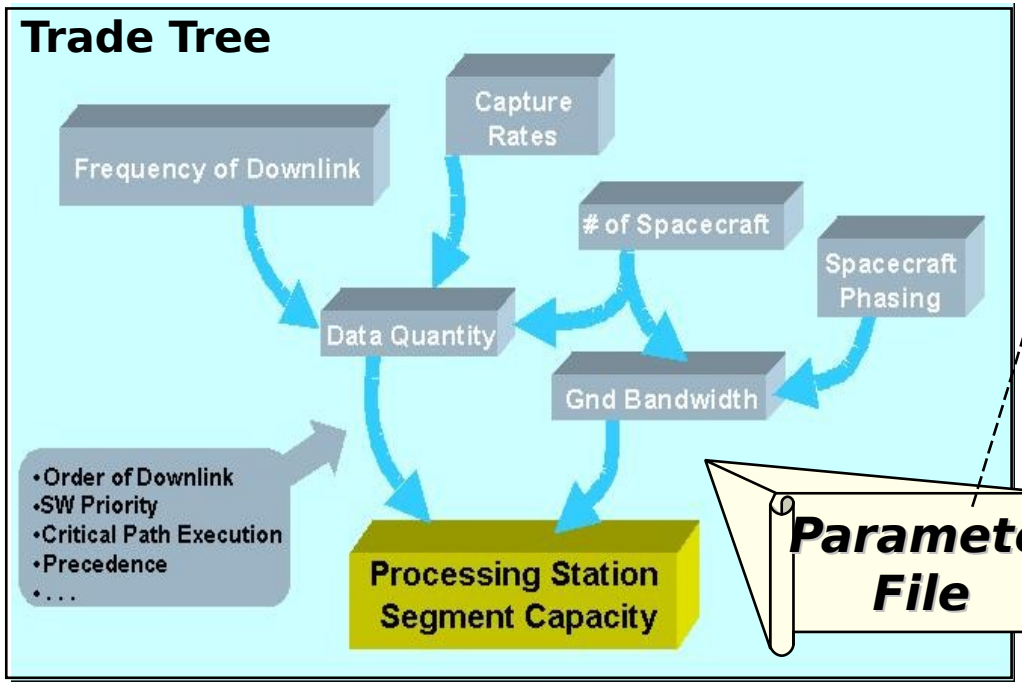
CI: Configuration Item

Agenda



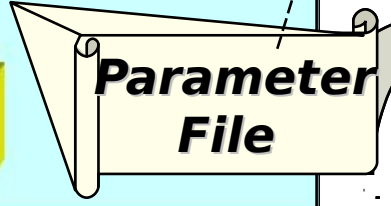
- **Modeling and Simulation**
- **System Trades**
- **Accurate Predictions**
- **Lifecycle Management / Control**
- **Lessons Learned**

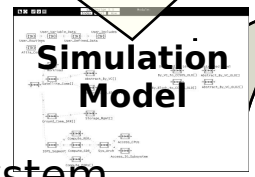
Performance is Influenced by the Interaction of the End-to-End System Components

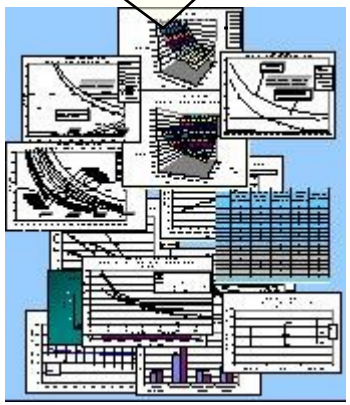


Parameter File approach:

- Decreases time to perform trade
  - Parameter file updates only
  - No re-compilation
  - No model re-verification
- Reduces risk
  - No model updates
  - No error introduction





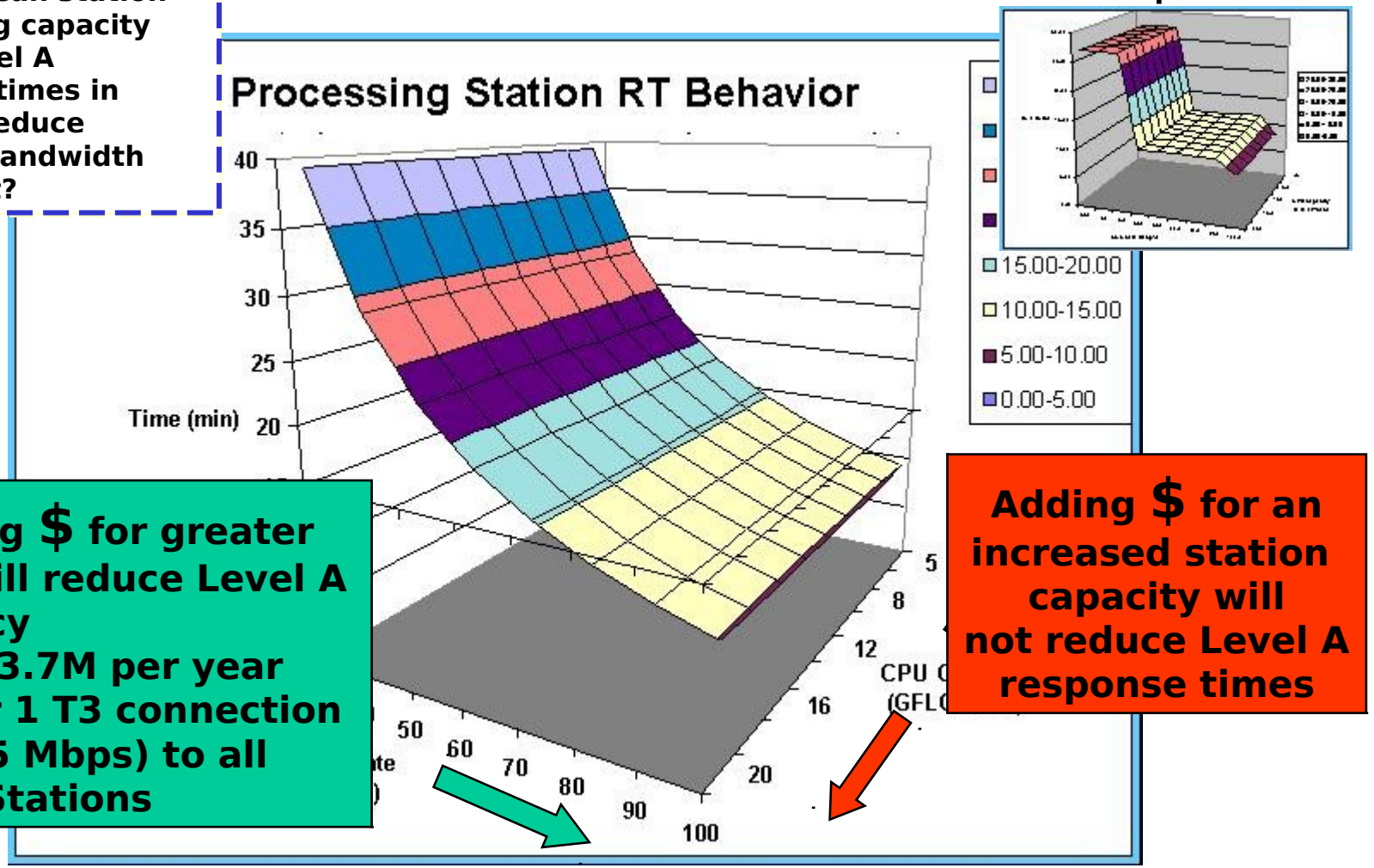


- Performance analysis and trade efforts to size a System Segment must consider the Segment as part of the larger system
- Segment capacity, performance behavior, and cost are influenced by the interaction of the other system components
- As requirements change, trades can be quickly updated for the generation of an optimal end-to-end system that meets requirements at minimal cost

Identify the Leverage Points that Improve System Performance

Purpose: **Can station processing capacity affect Level A response times in order to reduce network bandwidth (BW) cost?**

Per 45 Mbps Increments



Adding $ for greater BW will reduce Level A latency
- **$23.7M per year for 1 T3 connection (45 Mbps) to all 6 Stations**

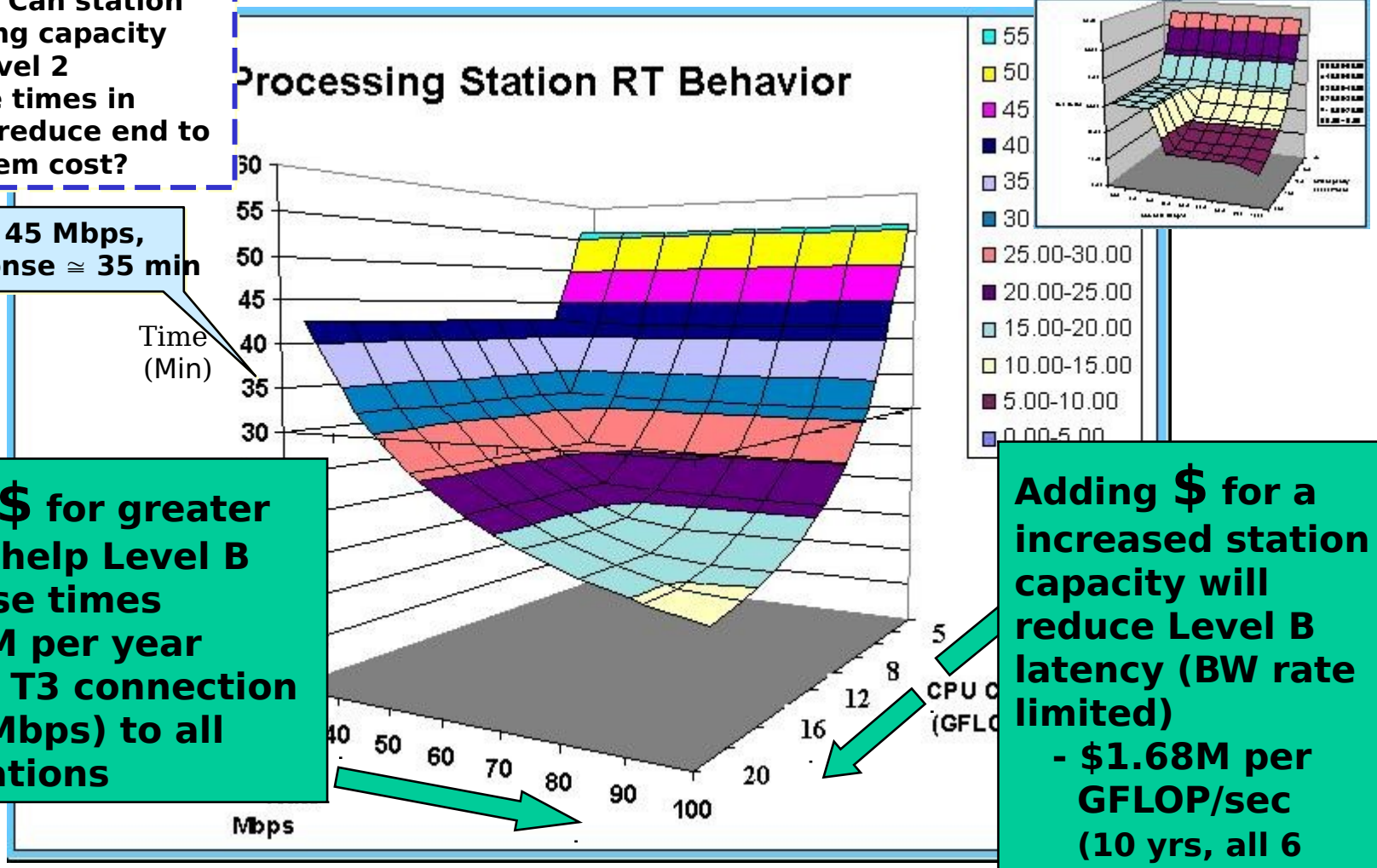
Adding $ for an increased station capacity will not reduce Level A response times

Understand the Interactions to Optimize Meeting Requirements and Overall Cost

Per 45 Mbps Increments

<u>Purpose:</u> Can station processing capacity affect Level 2 response times in order to reduce end to end system cost?

At 45 Mbps, response ≅ 35 min

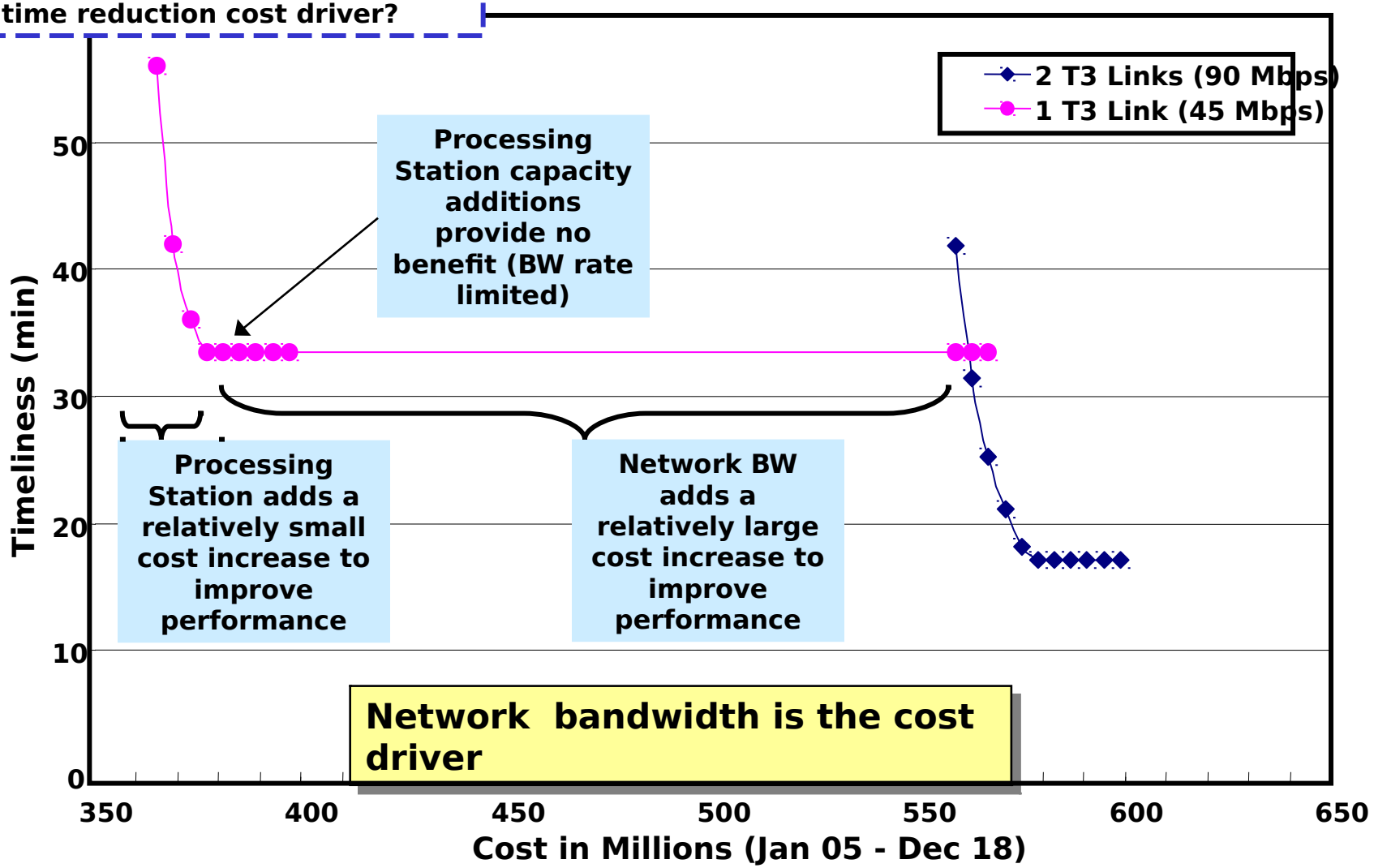


Adding $ for greater BW will help Level B Response times
- $23.7M per year for 1 T3 connection (45 Mbps) to all 6 Stations

Adding $ for a increased station capacity will reduce Level B latency (BW rate limited)
- $1.68M per GFLOP/sec (10 yrs, all 6 stations)

Determine the Cost Drivers of the Critical Performance Parameters

Purpose: What is the response time reduction cost driver?



Provide Alternatives to Minimize the Performance Cost Drivers

Purpose: **Can any system attributes be altered to minimize the response time cost driver?**

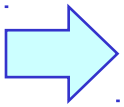
Base-line	Degrade Data (Lower Resolution)	Remove Protocol OH at Gnd Station	Change Com-pression Scheme	Limit Producer C's Data Xmission	Minimum Rate (Mbps)
1	0	0	0	0	72
1	0	0	1	0	65
1	0	1	0	0	61
1	0	1	1	0	56
1	1	0	0	0	55
1	0	0	0	1	54
1	1	0	1	0	48
1	0	0	1	1	47
1	1	1	0	0	46
1	0	1	0	1	45
1	1	1	1	0	41
1	0	1	1	1	39
1	1	0	0	1	38
1	1	0	1	1	35
1	1	1	0	1	34
1	1	1	1	1	30

Baseline

Configurations supporting a single T3 (45 Mbps)

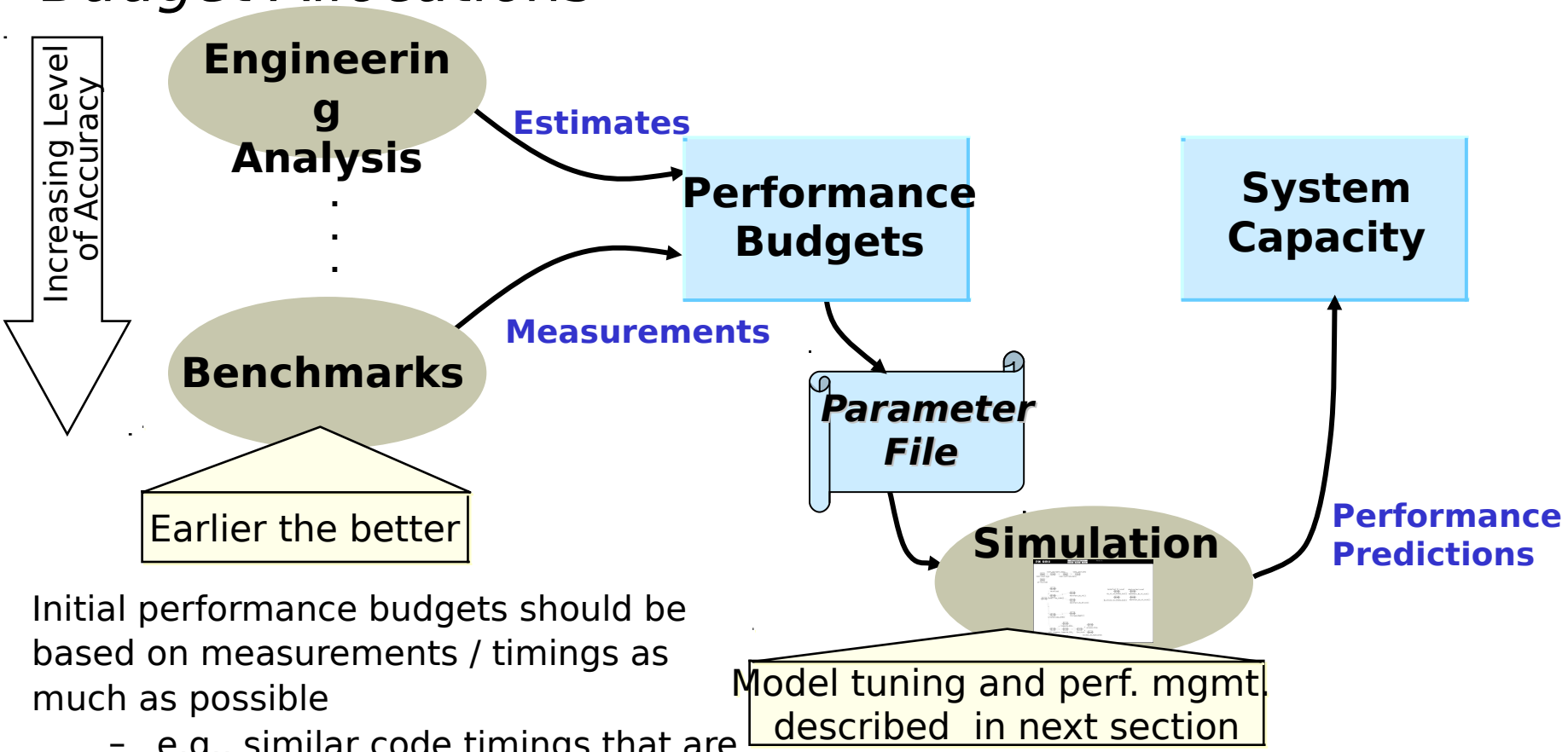
- **Saves $23.7M per year**

Agenda



- **Modeling and Simulation**
- **System Trades**
- **Accurate Predictions**
- **Lifecycle Management / Control**
- **Lessons Learned**

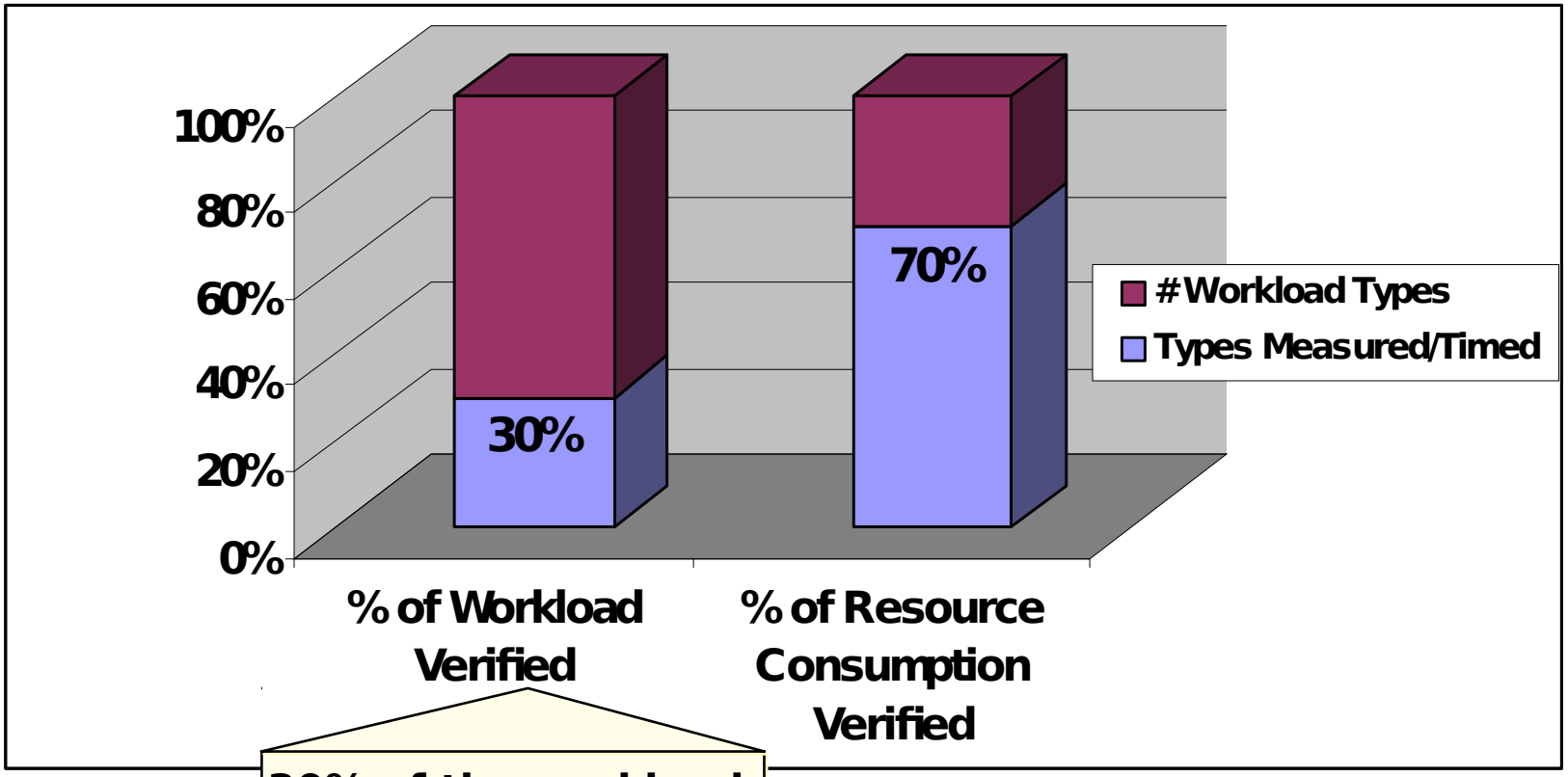
Timing Measurements Validate Budget Allocations



Initial performance budgets should be based on measurements / timings as much as possible

- e.g., similar code timings that are scaled by complexity factors
- e.g., application / code emulation or prototyping
- e.g., actual code execution

Develop and Benchmark the Performance Drivers First



30% of the workload consumes 70% of the system resources

This 30% Workload is Measured / Timed First

Account for all System Resource Consumption

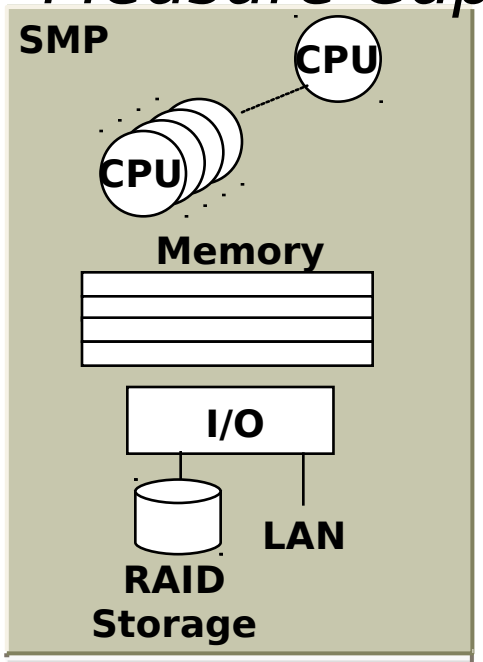
HARRIS

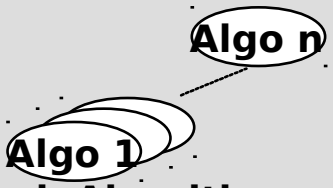
Do not forget about:

- Operating System
- Inter-process Communication
- System Management (SNMP Agents)
- Cluster Management
- Cluster File System
- Storage Management
- RBDMS
- Messaging
- Web Query Services
- FTP demons (Distribution)
- . . .

Create a Budget, even if it is a Percentage Overhead Tax

Emulate Execution for Quick Measure Capture





Each Algorithm:
- **Consumes CPU, Mem, and I/O BW**
- **Executes based on precedence rules**

```
. . .
for (i=1; i<=NUM_ALGOS; i++)
    if ((pid_a[i] = fork()) == 0)
    {
      execute_algos(i);
      exit(0);
    }
. . .
while (!precedence_met)
      if (semop(semid, &seize, 1) == ERROR)
. . .
memory = (char *)malloc(sizeof(char)*((int)
(1e6*mbytes)));
. . .
if ((t_result = pthread_create(&tid[j], &thread_attr,
          calc_mflop (void *) &split_mflops))!=0)
. . .
while (mflop > 0)
{
     d0value = d0value * d1value;
     d5value = d0value + d2value - d5value;
     d2value = d0value / d3value;
     mflop = mflop - 5.0;
}
. . .
for (i=start_block;i<=end_block;i++)
     fwrite((void *)the_data[i], sizeof(char),
               strlen(the_data[i]),data_file);
. . .
```

Create Processes

Control Execution Order

Consume Memory

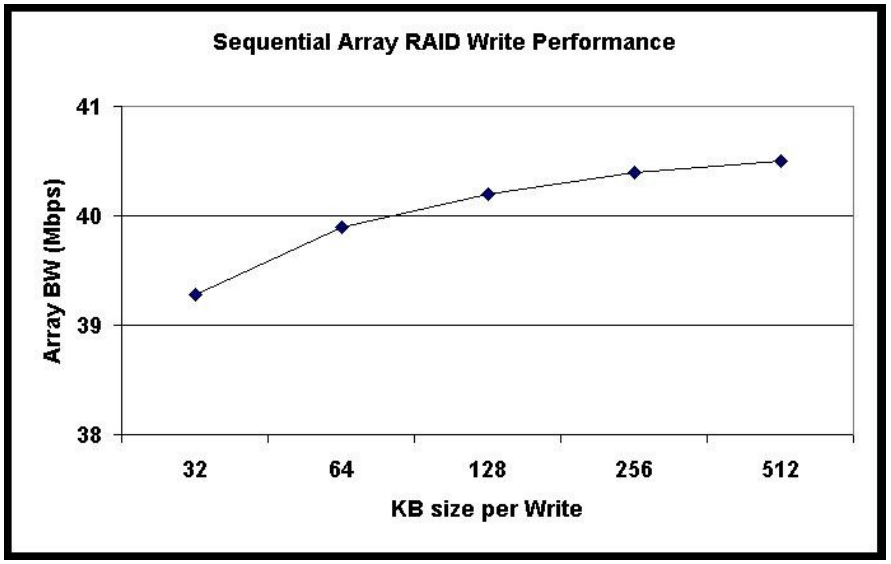
Create sub-Tasks

Consume CPUs

Consume I/O BW

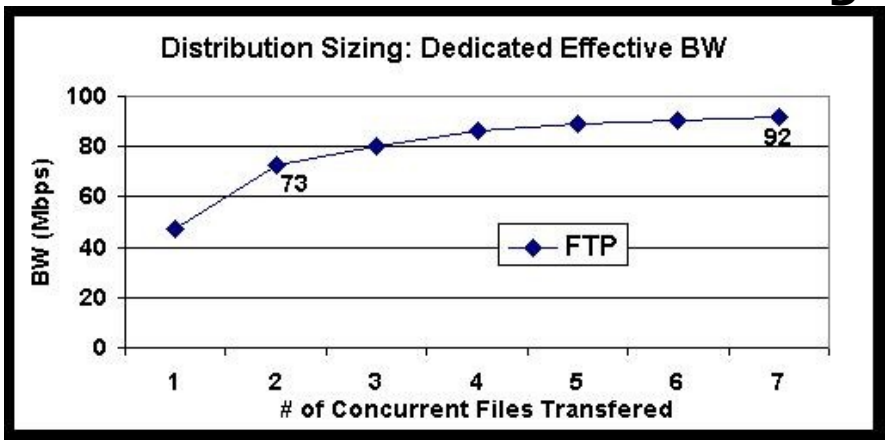
Benchmark for Discrete Measures and Performance Behavior

Dataset I/O BW Benchmarking



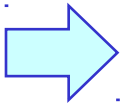
- RAID 5 configured disks to determine each trays effective write rate
- Two sets of timing tests:
  - sequential file writes
  - concurrent file writes
- File sizes and quantities for each test (based on dataset size distribution)
  - 10KB, 100KB, 500KB, 750KB, 1MB, 2MB, 4MB, 8MB, 16MB, 32MB each 1000 files
  - 64MB, 128MB, 256MB each 100 files
- 32KB to 512KB per write call

Dedicated Ethernet Benchmarking



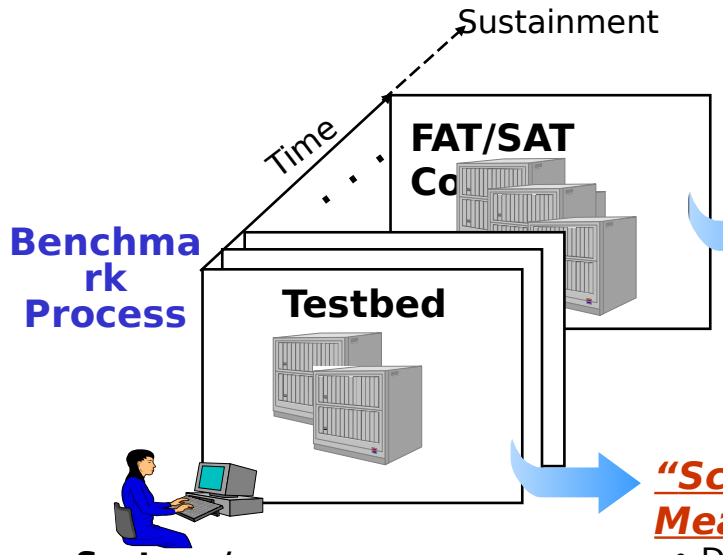
- Computer / disk to 100Mbps switch to computer / disk file transfers
- Sequential and concurrent file transfer utilized
- Each test consists of 10 runs transferring 5 100MB files each
- The concurrent file copies were performed by spawning off 5 separate processes that each simultaneously copied unique 100 MB files

Agenda



- **Modeling and Simulation**
- **System Trades**
- **Accurate Predictions**
- **Lifecycle Management / Control**
- **Lessons Learned**

Prediction, Measurement, and Management Process



- **Accurate Performance Predictions**
- **Program-Life Performance Assessment Control**

System Operational Performance Measurements

- Response Times
- Throughput
- Utilization
- . . .

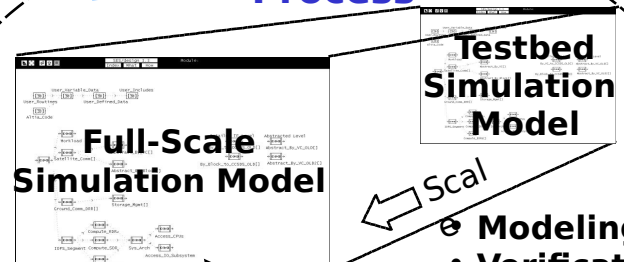
"Scalable" Performance Measurements

- Discrete
  - MFLOP
  - . . .
- Performance
  - Resp Times
  - . . .

"Technical" Performance Estimates

- MFLOPS / MIEs
- Logical / Phys. I/Os
- . . .

Simulation Process



- **Modeling**
- **Verification**
- **Analysis**
- **Tuning**

Assessment Process

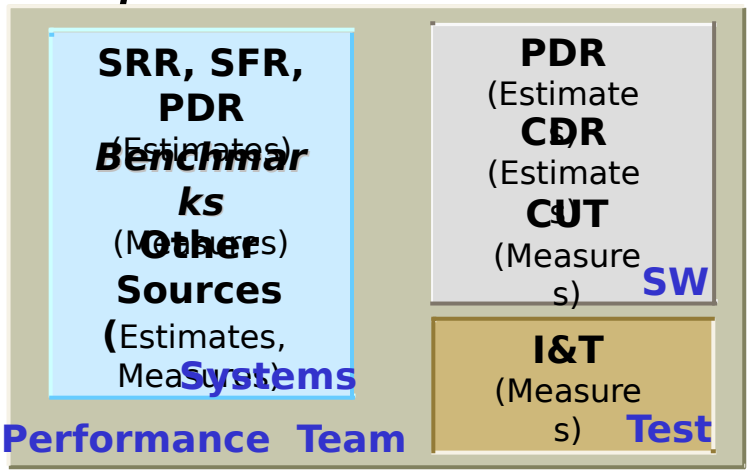
Discrete Budgets			Performance Budgets		
Allocation	Measure	Var %	Allocation	Measure	Var %

- **Estimation**
- **Verification**
- **Assessment**
- **Config Control**
- **Corrective Actions**

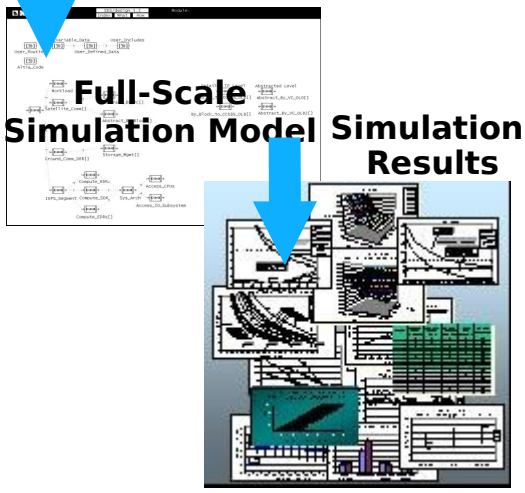
Performance Predictions

- Response Times
- Throughput
- . . .

At each Step, Ensure Requirements are Satisfied



1. Collect Estimates, Measures, Timings
   - Standard Engineer Processes
   - Organized Benchmarks
2. If < budget, hold as margin, stop
3. If > budget, assess impact



4. If minimal impact, use margin and reallocate budget, stop
5. If latency / perf req impact, begin recovery
   - Use margins
   - SW optimizations
   - System tuning
   - . . .
   - Increase HW capacity

Increasing Cost is last resort

- **Performance is continually assessed and managed by our process to verify that requirements are continually satisfied**
- **Our performance management process forces early issue resolution**

Manage Resource Consumption Budgets to Maintain the Baselined Capacity

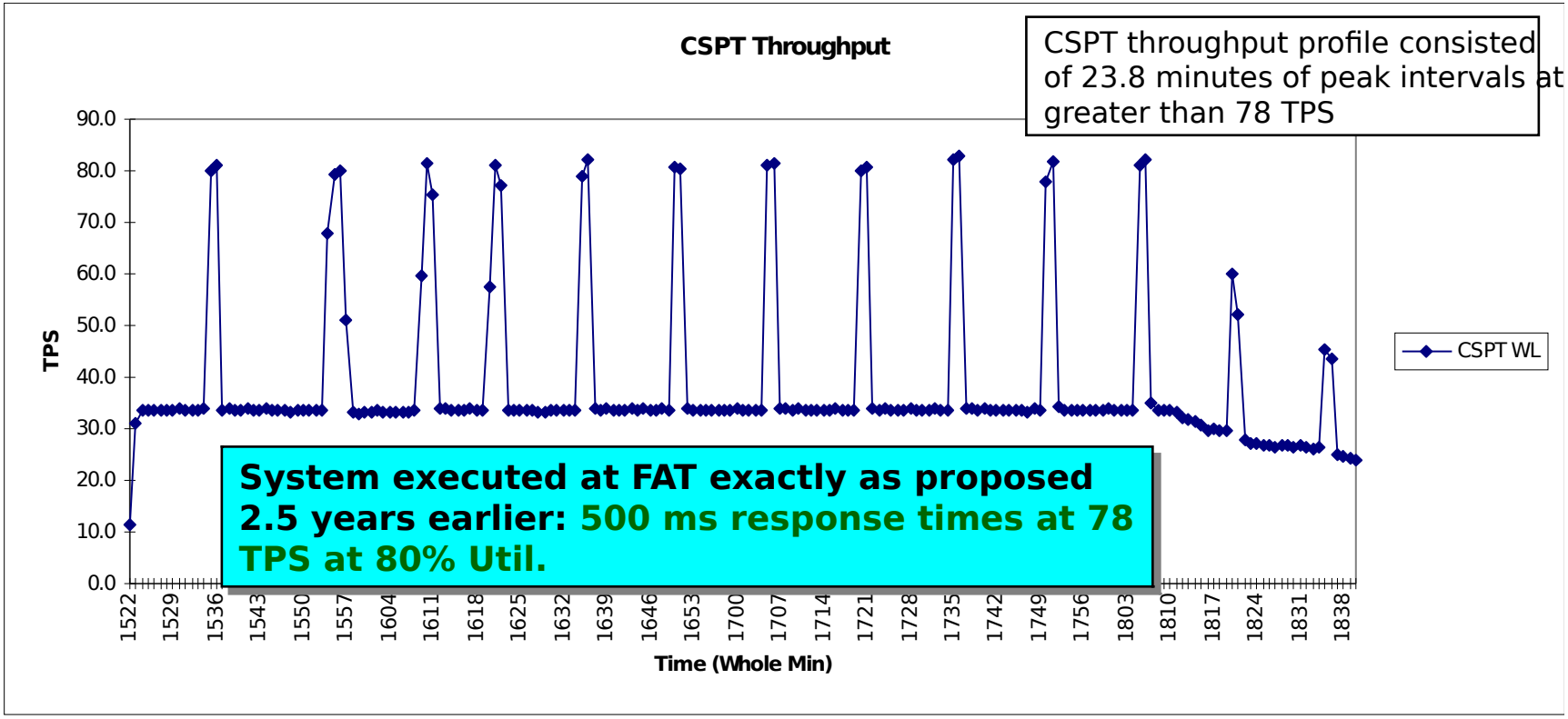
Algorithm	Budget / Measure Verification (CPU Consumption in MFLOPs)
Process A	+ 1.6X
Process B	+ 2.4X
Process C	+ 1.1X
Process D	+ 2.3X
Process E	+ 1.7X
Process F	+ 1.2X
Process G	- 2.1X

+ 1.03X

While +'s and –'s provide a balance, exercised "Optimization Modification" step of the "Algorithm SW Migration Process" to bring within budget / performance

HARRIS

Central Segment Performance Test System Throughput



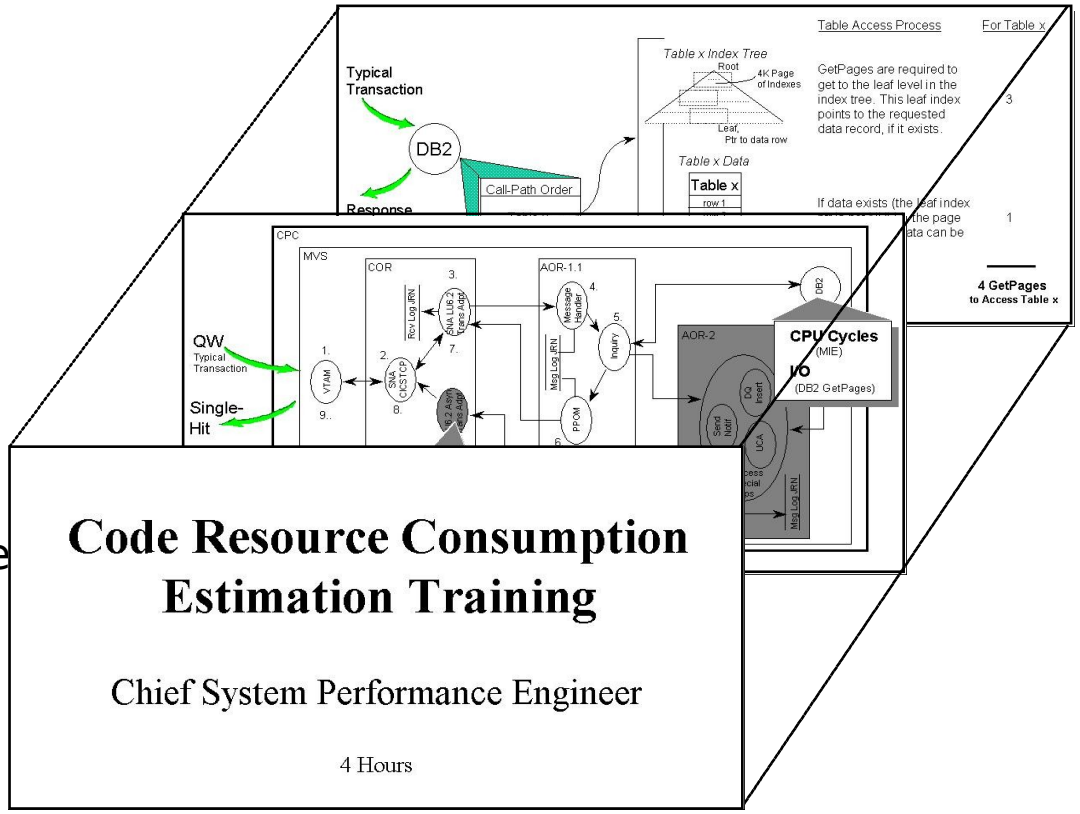
We have experience and have had extremely successful results using our performance assessment process

Estimate Resource Consumption Before Coding

Performance Estimation Training Class

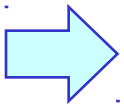
Topics

- Budget / Code Relationship
- Program Code Perf. Estimation Standard (CPES)
- Code MFLOP / MIE Estimation Template
- Code I/O Estimation Template
- Database MIE Estimation Template
- Database Logical I/O “Get-Page” Estimation Template
- . . .



Agenda

- **Modeling and Simulation**
- **System Trades**
- **Accurate Predictions**
- **Lifecycle Management / Control**
- **Lessons Learned**



Lessons Learned

- Get the Software organization involved in the performance assessment process as early as possible
- Have Chief Performance Engineer report directly to Program Manager on performance-critical systems
- Make sure the performance assessment activity is included in program cost and schedule
- Benchmark as early as possible
- Make sure the SW schedule is based on the performance drivers being developed first